


```

EEEEEEEEEE RRRRRRRR RRRRRRRR 000000 RRRRRRRR LL 000000 GGGGGGGG
EEEEEEEEEE RRRRRRRR RRRRRRRR 000000 RRRRRRRR LL 000000 GGGGGGGG
EE RR RR RR 00 00 RR RR LL 00 00 GG
EE RR RR RR 00 00 RR RR LL 00 00 GG
EE RR RR RR 00 00 RR RR LL 00 00 GG
EE RR RR RR 00 00 RR RR LL 00 00 GG
EEEEEEEE RRRRRRRR RRRRRRRR 00 00 RRRRRRRR LL 00 00 GG
EEEEEEEE RRRRRRRR RRRRRRRR 00 00 RRRRRRRR LL 00 00 GG
EE RR RR RR 00 00 RR RR LL 00 00 GG GGGGGG
EE RR RR RR 00 00 RR RR LL 00 00 GG GGGGGG
EE RR RR RR 00 00 RR RR LL 00 00 GG GG
EE RR RR RR 00 00 RR RR LL 00 00 GG GG
EEEEEEEEEE RR RR RR RR 000000 RR RR LLLLLLLL 000000 GGGGGG
EEEEEEEEEE RR RR RR RR 000000 RR RR LLLLLLLL 000000 GGGGGG

```

(1)	198	UNEXPECTED INTERRUPT SERVICE
(1)	315	LOG DEVICE ERRORS
(1)	407	LOG ASYNCHRONOUS DEVICE ATTENTIONS
(1)	492	LOG SOFTWARE STATUS
(1)	583	LOG DRIVER MESSAGE
(1)	644	ERL\$LOG DMSCP and ERL\$LOG TMSCP
(1)	703	BUILD STARTUP AND POWERFAIL MESSAGES
(1)	739	ALLOCATE ERROR MESSAGE BUFFER
(1)	803	GET FULL DEVICE NAME
(1)	844	RELEASE ERROR MESSAGE BUFFER
(1)	876	WAKE ERROR LOG FORMAT PROCESS

0000 1 .TITLE ERRORLOG - ERROR LOG SUPPORT ROUTINES
0000 2 :IDENT 'V04-000'
0000 3 *****
0000 4 *
0000 5 *
0000 6 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8 * ALL RIGHTS RESERVED.
0000 9 *
0000 10 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15 * TRANSFERRED.
0000 16 *
0000 17 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19 * CORPORATION.
0000 20 *
0000 21 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23 *
0000 24 *
0000 25 *****
0000 26
0000 27 D. N. CUTLER 7-MAR-77
0000 28
0000 29 ERROR LOG SUPPORT ROUTINES
0000 30
0000 31 MODIFIED BY:
0000 32
0000 33 V03-012 EAD0162 Elliott A. Drayton 26-Apr-1984
0000 34 Correct ADDB3 in routine GETFULLNAME to use R0.
0000 35
0000 36 V03-011 EAD0160 Elliott A. Drayton 16-Apr-1984
0000 37 Added a test for the system block address not being there.
0000 38
0000 39 V03-010 EAD0137 Elliott A. Drayton 11-Apr-1984
0000 40 Changed code to log full device names. NODE NAME + DEVICE.
0000 41
0000 42 V03-009 LMP0221 L. Mark Pilant, 30-Mar-1984 13:57
0000 43 Change UCBSL_OWNUIC to ORBSL_OWNER and UCBSW_VPROT to
0000 44 ORBSW_PROT.
0000 45
0000 46 V03-008 KPL0100 Peter Lieberwirth 22-Mar-1984
0000 47 Use CONFREGL instead of CONFREG. Anticipate SBICONF
0000 48 containing a PFN instead of a VA if BI adapter
0000 49 initialization didn't originally recognize the adapter.
0000 50
0000 51 V03-007 SSA0007 Stan Amway 2-Feb-1984
0000 52 Fix broken branch to ERLSALLOCUMB.
0000 53
0000 54 V03-006 LMP0185 L. Mark Pilant, 1-Feb-1984 9:37
0000 55 Fix some broken branches.
0000 56
0000 57 V03-005 ROW0241 Ralph O. Weber 12-OCT-1983

0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :-

Correct ERL\$LOG_D(T)MSCP to allocate space for the error log header in addition to the space needed for the logged message etc. Also use symbolic size of error log entry header instead of a constant.

V03-004 RLRMSCP Robert L. Rappaport 27-Jul-1983
Add two entrypoints, ERL\$LOG_DMSCP and ERL\$LOG_TMSCP, to log invalid Disk MSCP and Tape MSCP messages.

V03-003 KDM0051 Kathleen D. Morse 11-Jul-1983
Change references to TODR to use loadable, cpu-dependent routine, EXESREAD_TODR.

V03-002 BLS0187 Benn Schreiber 24-Sep-1982
Correct broken branch offset due to UCB growing

```
0000 75 :  
0000 76 : MACRO LIBRARY CALLS  
0000 77 :  
0000 78 :  
0000 79 :  
0000 80 :  
0000 81 :  
0000 82 :  
0000 83 :  
0000 84 :  
0000 85 :  
0000 86 :  
0000 87 :  
0000 88 :  
0000 89 :  
0000 90 :  
0000 91 :  
0000 92 :  
0000 93 :  
0000 94 :  
0000 95 :  
0000 96 :  
0000 97 :  
0000 98 :  
0000 99 :DEBUG=1  
0000 100 :  
0000 101 :  
0000 102 : LOCAL MACROS  
0000 103 :  
0000 104 :  
0000 105 :  
0000 106 :  
0000 107 : MACRO TO DEFINE AN INTERRUPT SERVICE ROUTINE LABEL FOR UNEXPECTED INTERRUPTS  
0000 108 :  
0000 109 :.MACRO ISRDEF,VNUM  
0000 110 :.ALIGN LONG  
0000 111 :ERL$VEC'VNUM:: : Make all vectors long word aligned  
0000 112 :.IF DF,DEBUG :INTERRUPT SERVICE LABEL  
0000 113 :BSBW ERL$UNEXP :***IF DEBUGGING  
0000 114 :.BYTE <VNUM>/2 :***CALL INTERRUPT SERVICE  
0000 115 :.ENDC :***IDENTIFY VECTOR OFFSET INTO SCB  
0000 116 :.ENDM ISRDEF  
0000 117 :  
0000 118 : MACRO TO DEFINE THE INTERRUPT SERVICE ROUTINE LABELS FOR AN ADAPTER  
0000 119 :  
0000 120 :.MACRO ADPISR,SLOT  
0000 121 :VECTOR = SLOT * 4 + 256  
0000 122 :.REPT 4  
0000 123 :.ISRDEF \VECTOR  
0000 124 :VECTOR = VECTOR + <16 * 4>  
0000 125 :.ENDR  
0000 126 :.IF NDF,DEBUG :IF NOT DEBUGGING  
0000 127 :BSBB ADP_HANDLER :CALL INTERRUPT SERVICE  
0000 128 :.ENDC  
0000 129 :.ENDM ADPISR  
0000 130 :  
0000 131 : LOCAL SYMBOLS
```

```

0000 132 :
0000 133 :
0000 134 :
0000 135 : MAXIMUM NUMBER OF MESSAGES BEFORE WAKE OF FORMAT PROCESS
0000 136 :
0000 137 :
0000 138 MAXMSG=10 ;
0000 139 :
0000 140 : MAXIMUM TIME IN SECONDS BEFORE WAKE OF FORMAT PROCESS
0000 141 :
0000 142 :
0000 143 :
0000 144 MAXTIM=30 ;
0000 145 :
0000 146 : LOCAL DATA
0000 147 :
0000 148 :
0000 149 :
0000 150 .PSECT $$S260,QUAD,WRT
0000 151 :
0000 152 : WARNING!!! The next two bytes must be adjacent and word aligned
0000 153 :
00 0000 154 .ALIGN WORD
00 0001 155 BUF1: .BYTE 0 ;COUNT OF BUSY MESSAGES IN BUFFER
00 0002 156 .BYTE 0 ;COUNT OF COMPLETED MESSAGES IN BUFFER
00 0003 157 .BYTE 0 ;BUFFER INDICATOR
0000000C 0004 158 .BYTE 0 ;BUFFER CONTROL FLAGS
00000200 0008 159 .LONG 10$ ;ADDRESS OF NEXT AVAILABLE SPACE IN BUFFER
00000200 000C 160 .LONG 20$ ;ADDRESS OF END OF BUFFER + 1
00000200 000C 161 10$: .BLKB 512-ERLSC_LENGTH ;ACTUAL BUFFER AREA
0200 162 20$: .REF LABEL ;REF LABEL
0200 163 :
0200 164 :
0200 165 : WARNING!!! The next two bytes must be adjacent and word aligned
0200 166 .ALIGN WORD
00 0200 167 BUF2: .BYTE 0 ;COUNT OF BUSY MESSAGES IN BUFFER
00 0201 168 .BYTE 0 ;COUNT OF COMPLETED MESSAGES IN BUFFER
01 0202 169 .BYTE 1 ;BUFFER INDICATOR
00 0203 170 .BYTE 0 ;BUFFER CONTROL FLAGS
0000020C 0204 171 .LONG 10$ ;ADDRESS OF NEXT AVAILABLE SPACE IN BUFFER
00000400 0208 172 .LONG 20$ ;ADDRESS OF END OF BUFFER + 1
00000400 020C 173 10$: .BLKB 512-ERLSC_LENGTH ;ACTUAL BUFFER AREA
0400 174 20$: .REF LABEL ;REF LABEL
0400 175 :
0400 176 : GLOBAL DATA
0400 177 :
0400 178 : ERROR LOG DATA BASE
0400 179 :
0400 180 :
0400 181 :
0400 182 ERLSAL_BUFADDR:: :ALLOCATION BUFFER ADDRESS ARRAY
00000000 0400 183 .LONG BUF1 :ADDRESS OF BUFFER 1 DESCRIPTOR
00000200 0404 184 .LONG BUF2 :ADDRESS OF BUFFER 2 DESCRIPTOR
0408 185 ERLSGB_BUFIND:: :CURRENT ALLOCATION BUFFER INDICATOR
00 0408 186 .BYTE 0 :
0409 187 ERLSGB_BUFFLAG:: :BUFFER STATUS FLAGS
00 0409 188 .BYTE 0 :

```

- ERROR LOG SUPPORT ROUTINES

M 4

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1Page 5
(1)

00	040A	189	ERL\$GB_BUFPTR::	:FORMAT PROCESS BUFFER INDICATOR
	040A	190	.BYTE 0	
	040B	191	ERL\$GB_BUFTIM::	:FORMAT PROCESS WAKEUP TIMER
1E	040B	192	.BYTE MAXTIM	
	040C	193	ERL\$GL_ERLPID::	:PROCESS ID OF ERROR LOG PROCESS
00000000	040C	194	.LONG 0	
	0410	195	ERL\$GL_SEQUENCE::	:UNIVERSAL ERROR SEQUENCE NUMBER
00000000	0410	196	.LONG 0	

0414 198 .SBTTL UNEXPECTED INTERRUPT SERVICE
 0414 199
 0414 200 ;+
 0414 201 : ERL\$VEC'VNUM - INTERRUPT SERVICE FOR SCB VECTOR VNUM.
 0414 202 : ERL\$UNEXP - GENERAL UNEXPECTED INTERRUPT SERVICE
 0414 203
 0414 204 : THESE INTERRUPT SERVICE ROUTINES ARE EXECUTED FOR UNUSED SCB VECTORS.
 0414 205
 0414 206 : IF DEBUG IS DEFINED, EACH INTERRUPT SERVICE CALLS ERL\$UNEXP WITH
 0414 207 : THE <VECTOR OFFSET>/2 INTO THE SCB AS A 1 BYTE ARGUMENT.
 0414 208
 0414 209 : IF DEBUG IS NOT DEFINED, ALL CPU INTERRUPT SERVICE ROUTINES COLLAPSE TO
 0414 210 : GLOBAL LABELS EQUAL TO ERL\$UNEXP AND ALL ADAPTER INTERRUPT SERVICE
 0414 211 : ROUTINES CALL A ROUTINE THAT SAVES THE ADAPTER TYPE, TRIES TO DISABLE
 0414 212 : FURTHER INTERRUPTS, AND LOGS THE INTERRUPT.
 0414 213
 0414 214 : THERE ARE ENOUGH INTERRUPT SERVICE ROUTINES FOR THE ARCHITECTURAL PAGE
 0414 215 : OF THE SCB, I.E., 128 ROUTINES.
 0414 216
 0414 217 : INPUTS:
 0414 218
 0414 219 : (SP) = PC AT INTERRUPT
 0414 220 : 4(SP) = PSL AT INTERRUPT
 0414 221
 0414 222 : OUTPUTS:
 0414 223
 0414 224 : ERROR IS LOGGED, OR PROCESSOR BUGCHECKS.
 0414 225 :-.
 00000000 226 :.PSECT \$AEXENONPAGED, LONG
 0000 227
 0000 228 : UNEXPECTED ADAPTER INTERRUPT HANDLER: IF DEBUG IS DISABLED, SAVE THE
 0000 229 : ADAPTER TYPE, ATTEMPT TO DISABLE FURTHER INTERRUPTS FROM THE ADAPTER,
 0000 230 : AND LOG THE INTERRUPT. IF DEBUG IS ENABLED, BUGCHECK AS FOR CPU INTERRUPTS.
 0000 231 :
 0000 232 :.ALIGN LONG
 0000 233 :ADP_UNEXP:
 0000 234 : NEXUS = 0 : FIRST ADAPTER = 0
 0000 235 :.REPT 16 : ISR'S FOR 16 ADAPTERS ONLY
 0000 236 :ADPISR \NEXUS : DEFINE ERL\$INT'VNUM LABELS AND ISRS
 0000 237 : NEXUS = NEXUS + 1 :NEXT ADAPTER
 0000 238 :.ENDR :
 0000 239 :ADP_HANDLER:
 003E 240 :SUBL #ADP_UNEXP+2,(SP) :COMPUTE ADAPTER OFFSET
 003E 241 :DIVL #4,(SP) :COMPUTE ADAPTER SLOT/TR NUMBER
 0045 242 :PUSHR #^M<R0,R1,R2,R3,R4> :SAVE REGISTERS
 0048 243 :MOVL 5+4(SP),R3 :RETRIEVE SLOT NUMBER
 004A 244 :MOVL @MMG\$GL_SBICONF[R3],R4 :GET ADDRESS OF ADAPTER REGISTERS
 004E 245 :BGEQ 100\$:GEQ MEANS SBICONF DOES NOT CONTAIN
 0056 246 : : A SYSTEM VA, MUST BE PFN OR 0
 0058 247
 0058 248 :SPRTCTINI B^5\$,#<MCHKSM_NEXM!MCHKSM_LOG>
 0064 249
 0064 250 :CLRL 4(R4) :DISABLE ADAPTER INTERRUPTS (HOPEFULLY)
 0067 251 :MOVL (R4),R1 :GET ADAPTER CONFIGURATION REG CONTENTS
 006A 252 :CMPB R1,#NDTS_DR32 :IS THIS A DR32?
 006D 253 :BNEQ 1\$:BRANCH IF NOT
 006F 254 :MOVL #^X500,(R4) :ELSE CLEAR INTERRUPTS IN SPECIAL WAY

```

54 64 D0 0076 255 1$: MOVL (R4),R4 ;GET THE ADAPTER'S CONFIGURATION REG
      0079 256
      0079 257 SPRCTEND 5$ ;IF R0 LBC, THEN NO ADAPTER PRESENT
      0079 258 BLBC R0,100$ ;IF NEQ, YES
37 50 E9 007A 259
      007D 260 TSTL @EXE$GL_CONFREGL[R3] ;ALREADY CONFIGURED?
      007D 261 BNEQ 10$ ;IF NEQ, YES
00000000'FF43 D5 007D 262 MOVZBL R4,@EXE$GL_CONFREGL[R3] ;SAVE THE ADAPTER TYPE
      08 12 0084 263 10$: ;SET SIZE OF MESSAGE TO ALLOCATE
      54 9A 0086 264 MOVL #EMBSC_UI_LENGTH,R1 ;ALLOCATE AN ERROR LOG BUFFER
      00000251'EF 18 D0 008E 265 JSB ERL$AL[OCEMB] ;BRANCH IF NONE AVAILABLE
      14 50 E9 0091 266 BLBC R0,20$ ;SET MESSAGE TYPE
      04 A2 0061 8F B0 009A 267 MOVW #EMBSC_UI_EMB$W_UI_ENTRY(R2) ;SET SLOT/TR NUMBER
      10 A2 53 D0 00A0 268 MOVL R3,EMBSL_UI_TR(R2) ;SET CONFIGURATION REGISTER VALUE
      14 A2 54 D0 00A4 269 MOVL R4,EMBSL_UI_CSR(R2) ;RELEASE BUFFER
      00000325'EF 16 00A8 270 JSB ERL$RELEASEMB ;RESTORE REGISTERS
      1F BA 00AE 271 20$: POPR #^M<R0,R1,R2,R3,R4> ;REMOVE SLOT NUMBER
      5E 04 C0 00B0 272 ADDL #4,SP
      02 00B3 273 REI
      00B4 274
      54 D4 0084 275 100$: CLRL R4 ;FLAG NO ADAPTER PRESETN
      D6 11 0086 276 BRB 10$ ;JOIN COMMON CODE
      00B8 277
      00B8 278 ; UNEXPECTED CPU INTERRUPT HANDLER: IF DEBUG IS ENABLED, BUGCHECK WITH
      00B8 279 ; <VECTOR OFFSET>/2 INTO SCB AS TOP BYTE ON STACK. IF DEBUG IS DISABLED,
      00B8 280 ; JUST BUGCHECK.
      00B8 281 ; JUST BUGCHECK.
      00B8 282 ; JUST BUGCHECK.
      00B8 283 ; ALIGN LONG
      00B8 284 CPU_UNEXP:
      00000000 00B8 285 VNUM=000 ;FIRST VECTOR = 0
      00B8 286 REPT 64 ;ISR'S FOR CPU INTERRUPTS ONLY
      00B8 287 ISRDEF \VNUM ;DEFINE ERL$INT'VNUM LABEL AND ISR
      00B8 288 VNUM=VNUM+4 ;NEXT VECTOR
      00B8 289 .ENDR
      00B8 290
      00B8 291 ERL$UNEXP:: ;***IF VECTOR ID ENABLED, ***
      00B8 292 .IF DF,DEBUG ;***OVERLAY RETURN WITH VECTOR OFFSET
      00B8 293 MOVZBL @(<SP>),<SP> ;***CONVERT ARG TO VECTOR OFFSET
      00B8 294 MULL #2,<SP>
      00B8 295 .IFTF
      00B8 296 BUG_CHECK UNXINTEXC ;BUGCHECK
      00B8 297 .IFT
      00B8 298 TSTL <SP>+
      00B8 299 .ENDC ;***CLEAN STACK
      02 00B8 300 REI ;RETURN FROM INTERRUPT
      00BD 301
      00BD 302 ; Vector entry for counting unexpected interrupts, rather than logging
      00BD 303 ; them. Used on 11/780 for passive release on the DW780 and for the
      00BD 304 ; CVTP microcode bug.
      00BD 305 ; .ALIGN LONG
      00CD 306
      00CD 307
      00CD 308
      00000000'EF D6 00C0 309 ERL$VEC_RETURN:: ; Increment counter
      02 00C6 310 INCL IOSGL_SCB_INTO ; And return
      00CD 311 REI

```

00C7 312
00C7 313

00C7 315 .SBTTL LOG DEVICE ERRORS
 00C7 316
 00C7 317 + ERL\$DEVICERR - LOG DEVICE CONTROLLER AND/OR DRIVE ERROR
 00C7 318 ERL\$DEVICTMO - LOG DEVICE TIMEOUT ERROR
 00C7 319
 00C7 320 THIS ROUTINE IS CALLED TO LOG A DEVICE TIMEOUT OR DEVICE CONTROLLER
 00C7 321 AND/OR DRIVE ERROR.
 00C7 322
 00C7 323
 00C7 324
 00C7 325
 00C7 326
 00C7 327
 00C7 328
 00C7 329
 00C7 330
 00C7 331
 00C7 332
 00C7 333
 00C7 334
 00C7 335
 00C7 336
 00C7 337 - ALL REGISTERS ARE PRESERVED ACROSS CALL.
 00000000 338 .PSECT WIONONPAGED
 0000 339 .ENABL LSB
 01 DD 0000 340 ERL\$DEVICERR:: :LOG DEVICE CONTROLLER AND/OR DRIVE ERROR
 05 11 0002 341 PUSHL #EMBSC_DE :SET FOR DEVICE ERROR
 0004 342 BRB 10\$:
 7E 0060 8F 3C 0004 343 ERL\$DEVICTMO:: :LOG DEVICE TIMEOUT ERROR
 03 38 A5 16 E0 0009 344 MOVZWL #EMBSC_DT,-(SP) :SET FOR DEVICE TIMEOUT
 0081 31 000E 345 10\$: BBS #DEVSV_ELG,UCBSL_DEVCHAR(R5),15\$:IF SET, ERROR LOG ENABLED
 F7 009A C5 0B E0 0011 346 12\$: BRW 40\$:ERROR LOG DISABLED
 0082 C5 B6 0017 347 15\$: BBS #IOSV_INNERLOG_UCBSW_FUNC(R5),12\$:IF SET, ERROR LOG INHIBITED
 72 64 A5 02 E0 001B 348 INCW UCBSW_ERRCNT(R5) :INCRÉMENT NUMBER OF DEVICE ERRORS
 004F 8F BB 0020 349 BBS #UCBSW_ERLOGIP_UCBSW_STS(R5),40\$:IF SET, ERROR IN PROGRESS
 53 28 A5 D0 0024 350 PUSHR #^M<R0,R1,R2,R3,R6> :SAVE REGISTERS
 56 0088 C5 D0 0028 351 MOVL UCBSL_DDB(R5),R5 :GET ADDRESS OF DDB
 51 16 A6 3C 002D 352 MOVL UCBSL_DDT(R5),R6 :GET ADDRESS OF DDT (from UCB not DDB)
 00000251'EF 16 0031 353 MOVZWL DDT\$W_ERRORBUF(R6),R1 :GET SIZE OF ERROR LOG BUFFER IN BYTES
 54 50 E9 0037 354 JSB ERL\$ALLOCEMB :ALLOCATE ERROR MESSAGE BUFFER
 0094 C5 52 D0 003A 355 BLBC R0,30\$:IF LBC ALLOCATION FAILURE
 64 A5 04 A8 003F 356 MOVL R2,UCBSL_EMB(R5) :SAVE ADDRESS OF ERROR MESSAGE BUFFER
 04 A2 14 AE B0 0043 357 BISW #UCBSM_ERLOGIP_UCBSW_STS(R5) :SIGNAL ERROR LOGGING IN PROGRESS
 52 1C C0 0048 358 MOVW 5*4(SPT,EMBSW_DV_ENTRY(R2)) :INSERT ENTRY TYPE
 0048 359 ADDL #EMBSB_DV_CLASS,R2 :POINT TO DEVICE CLASS
 0048 360
 82 40 A5 B0 004B 361 ASSUME EMBSB_DV_TYPE EQ EMBSB_DV_CLASS+1
 51 58 A5 D0 004B 362 MOVW UCBSB_DEVCLASS(R5),(R2)+ :INSERT DEVICE CLASS AND TYPE
 004F 363 MOVL UCBSL_IRP(R5),R1 :GET ADDRESS OF I/O PACKET
 0053 364
 82 0C A1 D0 0053 365 ASSUME EMBSL_DV_RQPID EQ EMBSB_DV_TYPE+1
 0053 366 MOVL IRPSL_PID(R1),(R2)+ :INSERT REQUESTER PROCESS ID
 0057 367
 0057 368 ASSUME EMBSW_DV_BOFF EQ EMBSL_DV_RQPID+4
 0057 369 ASSUME EMBSW_DV_BCNT EQ EMBSW_DV_BOFF+2
 82 30 A1 D0 0057 370 MOVL IRPSW_BOFF(R1),(R2)+ :INSERT TRANSFER PARAMETERS
 005B 371

82 00BC C5	DO	005B	372	ASSUME	EMBSL_DV MEDIA EQ	EMBSW DV BCNT+2
		0060	373	MOVL	UCBSL_MEDIA(R5),(R2)+	; INSERT SIZE OF DISK
82 54 A5	BO	0060	375	ASSUME	EMBSW_DV UNIT EQ	EMBSL DV MEDIA+4
		0064	376	MOVW	UCBSW_UNIT(R5),(R2)+	; INSERT UNIT NUMBER
82 0082 C5	BO	0064	378	ASSUME	EMBSW_DV ERRCNT EQ	EMBSW DV UNIT+2
		0069	379	MOVW	UCBSW_ERRCNT(R5),(R2)+	; INSERT NUMBER OF DEVICE ERRORS
82 70 A5	DO	0069	381	ASSUME	EMBSL_DV OPCNT EQ	EMBSW DV ERRCNT+2
		006D	382	MOVL	UCBSL_OPCT(R5),(R2)+	; INSERT OPERATIONS COMPLETED
50 1C A5	DO	006D	384	ASSUME	EMBSL_DV OWNUIC EQ	EMBSL DV OPCNT+4
82 60	DO	0071	385	MOVL	UCBSL_ORB(R5),R0	; GET ORB ADDRESS
		0074	386	MOVL	ORBSL_OWNER(R0),(R2)+	; INSERT VOLUME OWNER UIC
82 38 A5	DO	0074	388	ASSUME	EMBSL_DV CHAR EQ	EMBSL DV OWNUIC+4
		0078	389	MOVL	UCBSL_DEVCHAR(R5),(R2)+	; INSERT DEVICE CHARACTERISTICS
82 0090 C5	98	0078	391	ASSUME	EMBSB_DV SLAVE EQ	EMBSL DV CHAR+4
		007D	392	MOVZBW	UCBSB_SLAVE(R5),(R2)+	; INSERT SLAVE UNIT NUMBER
82 20 A1	BO	007D	394	ASSUME	EMBSW_DV FUNC EQ	EMBSB DV SLAVE+2
		0081	395	MOVW	IRPSW_FUNC(R1),(R2)+	; INSERT FUNCTION VALUE
7E 52 10	C1	0081	397	ASSUME	EMBST DV NAME EQ	EMBSW DV FUNC+2
0269 30	0085	398	ADDL3	#EMBSL_DV REGSAV-EMBST_DV_NAME,R2-(SP)	; CALCULATE ADDRESS OF REGIST	
50 8ED0	0088	399	BSBW	ERLSGETFULLNAME	; Copy full device name	
10 B6 16	008B	400	POPL	R0	; Restore address of register dump area	
004F 8F BA	008E	401	JSB	ADDTSL_REGDUMP(R6)	; CALL REGISTER DUMP ROUTINE	
5E 04 C0	0092	402	30\$:	POPR	#^M<R0,R1,R2,R3,R6>	; RESTORE REGISTERS
	0095	403	40\$:	ADDL2	#4,SP	; REMOVE ENTRY TYPE FROM STACK
	0096	404		RSB		
	0096	405		.DSABL	LSB	

.SBTTL LOG ASYNCHRONOUS DEVICE ATTENTIONS

0096 407
0096 408
0096 409 +
0096 410 ERL\$DEVICEATTN - Log asynchronous device attention interrupts that are
0096 411 not related to the current I/O operation that may be in progress.
0096 412
0096 413
0096 414
0096 415
0096 416
0096 417
0096 418
0096 419
0096 420
0096 421
0096 422
0096 423
0096 424
0096 425 -
0096 426
0096 427 ERL\$DEVICEATTN::
0096 428

0096 429
0096 430
0096 431
0096 432
0096 433
0096 434
0096 435
0096 436
0096 437
0096 438
0096 439
0096 440
0096 441
0096 442
0096 443
0096 444
0096 445
0096 446
0096 447
0096 448
0096 449
0096 450
0096 451
0096 452
0096 453
0096 454
0096 455
0096 456
0096 457
0096 458
0096 459
0096 460
0096 461
0096 462
0096 463

56 004F BF BB 0096	PUSHR #^M<R0,R1,R2,R3,R6>	; Save registers.	
51 0088 C5 D0 009A	MOVL UCBSL_DDT(R5),R6	; Get address of DDT.	
51 16 A6 3C 009F	MOVZWL DDT\$W-ERRORBUF(R6),R1	; R1=size of error log buffer in bytes.	
0082 C5 B6 00A3	INCW UCBSW_ERRCNT(R5)	; Increment number of device errors.	
51 16 E1 00A7	BBC #DEVSV ELG,-		
5D 38 A5 00A9	UCBSL_DEVCHAR(R5),30\$		
01A2 30 00AC	BSBW ERL\$A[LOCEMB	; If clr, error log disabled.	
57 50 E9 00AF	BLBC R0,30\$; Allocate error message buffer.	
52 DD 00B2	PUSHL R2	; If LBC allocation failure.	
0062 8F B0 00B4	MOVW #EMBS\$C_DA,-	; Save address of allocated buffer.	
04 A2 00B8	EMBSW_DV_ENTRY(R2)		
64 A5 B0 00BA	MOVW UCB\$W_STS(R5),-	; Insert entry type.	
1A A2 00BD	EMBSW_DV_STS(R2)	; Save device status in buffer.	
12 A2 7C 00BF	CLRQ EMBSQ_DV_IOSB(R2)		
52 1C C0 00C2	ADDL #EMBSB_DV_CLASS,R2	; Clear irrelevant field.	
52 1C C0 00C5		; R2 => device class field.	
82 40 A5 B0 00C5	ASSUME EMB\$B_DV_TYPE EQ	EMBSB_DV_CLASS+1	
	MOVW UCB\$B_DEVCLASS(R5),(R2)+	; Insert device class and type.	
	00C9 446		
	00C9 447		
	00C9 448		
	00C9 449	ASSUME EMB\$L_DV_RQPID EQ	EMBSB_DV_TYPE+1
	00C9 450	ASSUME EMB\$W_DV_BOFF EQ	EMBSL_DV_RQPID+4
	00C9 451	ASSUME EMB\$W_DV_BCNT EQ	EMBSW_DV_BOFF+2
82 7C 00C9	CLRQ (R2)+	; Clear PID, BOFF and BCNT.	
	00CB 452		
	00CB 453		
82 00BC C5 D0 00CB	ASSUME EMB\$L_DV_MEDIA EQ	EMBSL_DV_BCNT+2	
	00CB 454	MOVL UCBSL_MEDIA(R5),(R2)+	; Insert size of disk.
	00D0 455		
	00D0 456		
82 54 A5 B0 00D0	ASSUME EMB\$W_DV_UNIT EQ	EMBSL_DV_MEDIA+4	
	00D0 457	MOVW UCB\$W_UNIT(R5),(R2)+	; Insert unit number.
	00D4 458		
	00D4 459		
82 0082 C5 B0 00D4	ASSUME EMB\$W_DV_ERRCNT EQ	EMBSW_DV_UNIT+2	
	00D4 460	MOVW UCB\$W_ERRCNT(R5),(R2)+	; Insert number of device errors.
	00D9 461		
	00D9 462		
	00D9 463	ASSUME EMB\$L_DV_OPCNT EQ	EMBSW_DV_ERRCNT+2

010E 492 .SBTTL LOG SOFTWARE STATUS
 010E 493
 010E 494
 010E 495 ;+ ERL\$LOGSTATUS - Log software status corresponding to a logged message.
 010E 496
 010E 497 ; INPUTS:
 010E 498 R0-R1 contain final I/O status
 010E 499 R2 => MSCP end message
 010E 500 R3 => UCB
 010E 501 R5 => CDRP
 010E 502
 010E 503 ; OUTPUTS:
 010E 504 An error log message (format EMBSPDEF) is allocated and filled in.
 010E 505 All registers are preserved.
 010E 506
 010E 507
 010E 508 ERL\$LOGSTATUS::
 010E 509
 0082 C3 B6 010E 510 INCW UCBSW_ERRCNT(R3) ; Increment number of device errors.
 16 E1 0112 511 BBC #DEV\$0 ELG,- ; If clear, error log disabled.
 71 38 A3 0114 512
 0117 513
 7E 50 7D 0117 514 MOVO R0,-(SP) ; Save R0, R1, R2.
 52 DD 011A 515 PUSHL R2
 51 0050 8F 3C 011C 516 MOVZWL #EMBSK_SP_LENGTH,R1 ; R1 contains length of buffer to alloc
 012D 30 0121 517 BSBW ERL\$ALLOCEMB ; Allocate error message buffer.
 5B 50 E9 0126 518 BLBC R0,10\$; LBC implies allocation failure.
 0063 BF B0 0127 519 MOVW #EMBSB_SP_- ; Indicate type of error log buffer.
 04 A2 0128 520 EMB\$W_SP_ENTRY(R2)
 50 10 A2 9E 012D 521 MOVAB EMB\$B_SP_CLASS(R2),R0 ; R0 => where to begin filling.
 0131 522
 0131 523 ASSUME UCBSB_DEVTYPE EQ UCBSB_DEVCLASS+1
 0131 524 ASSUME EMB\$B_SP_TYPE EQ EMB\$B_SP_CLASS+1
 80 40 A3 B0 0131 525 MOVW UCBSB_DEVCLASS(R3),(R0)+ ; Movē Device type and class.
 0135 526
 0135 527 ASSUME EMB\$W_SP_BOFF EQ EMB\$B_SP_TYPE+1
 80 D0 A5 B0 0135 528 MOVW CDRPSQ_BOFF(R5),(R0)+ ; Copy BOFF.
 0139 529
 0139 530 ASSUME EMB\$L_SP_BCNT EQ EMB\$W_SP_BOFF+2
 80 D2 A5 D0 0139 531 MOVL CDRPSL_BCNT(R5),(R0)+ ; Also byte count.
 0130 532
 0130 533 ASSUME EMB\$L_SP_MEDIA EQ EMB\$L_SP_BCNT+4
 80 DB A5 D0 0130 534 MOVL CDRPSL_MEDIA(R5),(R0)+ ; Movē media address (LBN).
 0141 535
 0141 536 ASSUME EMB\$L_SP_RQPID EQ EMB\$L_SP_MEDIA+4
 80 AC A5 D0 0141 537 MOVL CDRPSL_PID(R5),(R0)+ ; Copy requesting PID.
 0145 538
 0145 539 ASSUME EMB\$Q_SP_IOSB EQ EMB\$L_SP_RQPID+4
 80 04 AE 7D 0145 540 MOVO 4(SP),(R0)+ ; Copy saved I/O status to buffer.
 0149 541
 0149 542 ASSUME EMB\$W_SP_FUNC EQ EMB\$Q_SP_IOSB+8
 80 C0 A5 B0 0149 543 MOVW CDRPSQ_FUNC(R5),(R0)+ ; Copy I70 function code.
 014D 544
 014D 545 ASSUME EMB\$W_SP_UNIT EQ EMB\$W_SP_FUNC+2
 80 54 A3 B0 014D 546 MOVW UCBSW_UNIT(R3),(R0)+ ; Copy unit number.
 0151 547
 0151 548

80 70 A3 00	0151	549	ASSUME MOVL	EMBSL_SP_OPCNT EQ UCBSL_OPENT(R3),(R0)+	EMBSW_SP_UNIT+2 ; Copy cūmulative operation count.
80 0082 C3 B0	0155	550	ASSUME MOVW	EMBSW_SP_ERRCNT EQ UCBSW_ERRCNT(R3),(R0)+	EMBSL_SP_OPCNT+4 ; And also cūmulative error count.
80 64 A3 B0	015A	551	ASSUME MOVW	EMBSW_SP_UCBSTS EQ UCBSW_STS(R3),(R0)+	EMBSW_SP_ERRCNT+2 ; Copy UCB STS field.
51 1C A3 D0	015E	552	ASSUME MOVL	EMBSL_SP_OWNUIC EQ UCBSL_ORB(R3),R1	EMBSW_SP_UCBSTS+2 ; GET ORB ADDRESS
80 61 D0	0162	553	MOVL	ORBSSL_OWNER(R1),(R0)+	; Copy device owner UIC.
80 38 A3 D0	0165	554	ASSUME MOVL	EMBSL_SP_CHAR EQ UCBSL_DEVCHAR(R3),(R0)+	EMBSL_SP_OWNUIC+4 ; Copy device characteristics.
51 6E D0	0169	555	ASSUME MOVL	EMBSL_SP_CMDREF EQ (SP),R1	EMBSL_SP_CHAR+4 ; R1 => MSCP end message.
80 61 D0	016C	556	MOVL	MSCP\$L_CMD_REF(R1),(R0)+	; Copy command reference number (RSPID).
52 52 7D	016F	557	ASSUME MOVQ	EMBSL_SP_DEVNAM EQ R2,-(SP)	EMBSL_SP_CMDREF+4 ; Save UCB & buffer base address(R2,R3)
52 50 D0	0172	558	MOVL	R0,R2	; Get buffer address
53 28 A3 D0	0175	559	MOVL	UCBSL_DDB(R3),R3	; Get DDB address
0175 30	0179	560	BSBW	ERL\$GETFULLNAME	; Copy full device name
52 8E 7D	017C	561	MOVQ	(SP)+,R2	; Restore R2 and R3
01A3 30	017F	562	BSBW	ERL\$RELEASEMB	; Release filled in error buffer.
50 52 8E D0	0182	563	10\$:	POPL R2	
50 8E 7D	0185	564	MOVQ (SP)+,R0		; Restore registers R2, R1, R0.
05 0188	0188	565	20\$:	RSB	
		581			; Return to caller.

0189 583 .SBTTL LOG DRIVER MESSAGE

0189 584

0189 585 ;+ ERL\$LOGMESSAGE - Subroutine to allocate a message buffer, fill in a standard header, and then copy caller specified text to the rest of the buffer.

0189 586

0189 587

0189 588

0189 589

0189 590 INPUTS:
R0 = Code specifying message sub type.
R1 = Length of caller specified text
R2 => caller text
R3 => UCB

0189 591

0189 592

0189 593

0189 594

0189 595 ; OUTPUTS:
Message allocated and filled. All registers preserved.

0189 596

0189 597

0189 598

0189 599

0189 600 ERL\$LOGMESSAGE::

0189 601

0082 C3 B6 0189 602 INCW UCB\$W_ERRCNT(R3) ; Increment total number of errors.
16 E1 018D 603 BBC #DEV\$V_ELG,- ; Clear means error logging inhibited.

55 38 A3 018F 604

7E 50 7D 0192 605

7E 52 7D 0195 606

7E 54 7D 0198 607

51 26 C0 019B 608

0080 30 019E 609

3A 50 E9 01A1 610

52 DD 01A4 611

0064 8F B0 01A6 612

04 A2 01AA 613

01AC 614

01AC 615

01AC 616

01AC 617

01AC 618

40 A3 B0 01AC 619

10 A2 01AF 620

01B1 621

54 A3 B0 01B1 622

12 A2 01B4 623

01B6 624

53 DD 01B6 625

52 14 A2 DE 01B8 626

28 A3 D0 01BC 627

012E 30 01C0 628

53 8ED0 01C3 629

52 6E D0 01C6 630

01C9 631

24 A2 14 AE B0 01C9 632

51 0C AE D0 01CE 633

26 A2 61 18 AE 28 01D2 634

52 8ED0 01D8 635

0147 30 01DA 636

01DE 637 10\$:

01DE 638

54 8E 7D 01E1 639

0189 625

0189 626

0189 627

0189 628

0189 629

0189 630

0189 631

0189 632

0189 633

0189 634

0189 635

0189 636

0189 637

0189 638

0189 639

INCL UCB\$W_ERRCNT(R3) ; Increment total number of errors.
BBC #DEV\$V_ELG,- ; Clear means error logging inhibited.

MOVQ R0,-(SP) ; Save registers R0-R5.

MOVQ R2,-(SP)

MOVQ R4,-(SP)

ADDL #EMBSK_LM_LENGTH,R1 ; Add message header to caller's length.

BSBW ERL\$ALLOCATEMB ; Allocate buffer.

BLBC R0,10\$; LBC means allocation failure.

PUSHL R2 ; Save address of buffer.

MOVW #EMBSL_LM_- ; Indicate type of error log buffer.

EMBSL_LM_ÉNTRY(R2)

ASSUME UCB\$B_DEVTYPE EQ ; Device type and class.

ASSUME EMB\$B_LM_TYPE EQ

MOVW UCB\$B_DEVCLASS(R3),- ; Begin to fill in buffer. Copy

EMBS\$B_LM_CLASS(R2)

MOVW UCB\$W_UNIT(R3),- ; Also copy device unit number.

EMBS\$W_LM_UNIT(R2)

PUSHL R3 ; Save UCB

MOVAL EMB\$T_LM_DEVNAM(R2),R2 ; Get buffer address for device name

MOVL UCB\$L_DDB(R3),R3 ; Get DDB address

BSBW ERL\$GETFULLNAME ; Copy full device name

POPL R3 ; Restore UCB

MOVL (SP),R2 ; Restore buffer base address

MOVW 20(SP),EMBS\$W_LM_MSGTYP(R2) ; Copy message subtype.

MOVL 12(SP),R1 ; R1 => caller's text.

MOVC3 24(SP),(R1),EMBS\$W_LM_MSGTYP+2(R2) ; Copy caller's text.

POPL R2 ; R2 => allocated buffer.

BSBW ERL\$RELEASEMB ; Release buffer.

MOVQ (SP)+,R4 ; Restore Registers R0-R5.

MOVQ (SP)+,R2

50 8E 7D 01E6 640 20\$: MOVQ (SP)+,R0
05 01E7 641 RSB

; Return to caller.

01E8 644 .SBTTL ERL\$LOG_DMSCP and ERL\$LOG_TMSCP
 01E8 645
 01E8 646
 01E8 647 :+ Routines that respectively log invalid Disk and Tape MSCP messages.
 01E8 648
 01E8 649 Inputs:
 01E8 650 R0 = type of message
 01E8 651 R1 = length of message
 01E8 652 R2 => message
 01E8 653 R3 => CDBB
 01E8 654
 01E8 655 Outputs:
 01E8 656 All registers preserved.
 01E8 657
 01E8 658 We want to log the following items in addition to the message
 01E8 659 and its type:
 01E8 660
 01E8 661 1. CDBBSB_SYSTEMID (6 bytes)
 01E8 662 2. The ASCII string "DISK" (4 bytes) or "TAPE" (4 bytes)
 01E8 663 3. CDBBSQ_CNTRLID (8 bytes)
 01E8 664
 01E8 665 .enabl lsb
 01E8 666
 01E8 667
 01E8 668 ERL\$LOG_TMSCP:::
 01E8 669
 54 45504154 3F BB 01E8 670 PUSHR #^M<R0,R1,R2,R3,R4,R5> ; Save registers.
 8F D0 01EA 671 MOVL #^A/TAPE/,R4 ; R4 has string "TAPE".
 09 11 01F1 672 BRB 10\$; Branch around to common code.
 01F3 673 ERL\$LOG_DMSCP:::
 01F3 674
 54 48534944 3F BB 01F3 675 PUSHR #^M<R0,R1,R2,R3,R4,R5> ; Save registers.
 8F D0 01F5 676 MOVL #^A/DISK/,R4 ; R4 has string "DISK".
 51 24 C0 01FC 677 10\$: ADDL #<2+4+6+8 -
 01FF 678 +EMBSK_HD_LENGTH>,R1 ; R1 has length which is bumped by
 01FF 679 ; 2 for the type, 4 for 'DISK' or
 01FF 680 ; "TAPE", 6 for SYSTEMID, 8 for
 01FF 681 ; CNTRLID, and errorlog entry header
 01FF 682 ; size.
 004F 30 01FF 683 BSBW ERL\$ALLOCEMB ; Allocate Errorlog Buffer.
 27 50 E9 0202 684 BLBC R0,20\$; LBC means no allocate.
 52 DD 0205 685 PUSHL R2 ; Save R2=>Buffer.
 0065 8F B0 0207 686 MOVW #EMBSCLOGMSCP -
 04 A2 0208 687 EMBSW RD ENTRY(R2) ; Copy message class to buffer header.
 52 10 C0 020D 688 ADDL #EMBSK_HD_LENGTH,R2 ; R2 => beyond header.
 82 04 AE B0 0210 689 MOVW 4(SP),R2+ ; Copy message type (from saved regs).
 82 54 D0 0214 690 MOVL R4,(R2)+ ; Copy Class driver type.
 82 20 A3 70 0217 691 MOVQ CDDBSQ_CNTRLID(R3),(R2)+ ; Controller identifier.
 82 0C A3 70 021B 692 MOVQ CDDBSB_SYSTEMID(R3),(R2)+ ; And System ID.
 08 AE 28 021F 693 MOVC3 8(SP) = ; Get length from saved registers.
 0C BE 0222 694 ; also source address.
 FE A2 0224 695
 52 8ED0 0226 696 POPL R2 ; Target is -2 since SYSTEMID is 6 bytes.
 00F9 30 0229 697 BSBW ERL\$RELEASEMB ; Restore R2=>Buffer.
 3F BA 022C 698 20\$: POPR RSB ; Free Errorlog buffer.
 05 022E 700

ERRORLOG
V04-000

- ERROR LOG SUPPORT ROUTINES
ERL\$LOG_DMSCP and ERL\$LOG_TMSCP

M 5

022F 701

.dsabl lsb

16-SEP-1984 00:04:39 VAX/VMS Macro V04-00
5-SEP-1984 03:41:34 [SYS.SRC]ERRORLOG.MAR;1

Page 18
(1)

EXI
VO

0251 739 .SBTTL ALLOCATE ERROR MESSAGE BUFFER
 0251 740
 0251 741 : ERL\$ALLOCMB - ALLOCATE ERROR MESSAGE BUFFER
 0251 742 : THIS ROUTINE IS CALLED TO ALLOCATE AN ERROR LOG MESSAGE BUFFER AND
 0251 743 : INITIALIZE ITS HEADER.
 0251 744 :
 0251 745 : INPUTS:
 0251 746 :
 0251 747 : R1 = SIZE OF MESSAGE BUFFER REQUIRED IN BYTES.
 0251 748 :
 0251 749 :
 0251 750 : OUTPUTS:
 0251 751 :
 0251 752 : R0 LOW BIT CLEAR INDICATES AN ALLOCATION FAILURE.
 0251 753 :
 0251 754 : R0 LOW BIT SET INDICATES SUCCESSFUL ALLOCATION WITH:
 0251 755 :
 0251 756 : R1 = ERROR SEQUENCE NUMBER.
 0251 757 : R2 = ADDRESS OF ALLOCATED ERROR MESSAGE BUFFER.
 0251 758 :
 0251 759 : IN EITHER CASE THE UNIVERSAL ERROR SEQUENCE NUMBER IS INCREMENTED
 0251 760 : AND STORED IN THE BUFFER AT THE STANDARD PLACE, ALONG WITH THE TIME.
 0251 761 : AND THE ERROR LOG PROCESS MAY BE AWAKENED IF AN ERROR ALLOCATION
 0251 762 : BUFFER IS FOUND TO BE FULL.
 0251 763 :
 0251 764 : R3 IS PRESERVED ACROSS CALL.
 0251 765 :
 0251 766 :
 0251 767 ERL\$ALLOCMB:: : ALLOCATE ERROR MESSAGE BUFFER
 0251 768 DSBINT : DISABLE ALL INTERRUPTS
 50 51 04 C0 0257 769 ADDL #EMBSK_LENGTH,R1 : Add in size of header for message
 50 00000408'EF 9A 025A 770 MOVZBL ERL\$GB_BUFIND,R0 : GET CURRENT ALLOCATION BUFFER INDICATOR
 1E 03 A0 00 D0 0261 771 MOVL ERL\$AL_BUFADDR[R0],R0 : GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
 04 A0 52 51 C1 026E 772 BBS #ERLSV_LOCK,ERLSB_FLAGS(R0) : 15\$: IF SET, BUFFER BEING COPIED
 04 A0 52 51 C1 0272 773 MOVL ERL\$L_NEXT(R0),R2 : GET ADDRESS OF NEXT AVAILABLE SPACE
 04 A0 08 A0 D1 0277 774 ADDL3 R1,R2,ERL\$L_NEXT(R0) : CALCULATE ADDRESS OF NEXT AVAILABLE SPACE
 42 1E 027C 775 CMPL ERL\$L_END(R0),ERL\$L_NEXT(R0) : ENTRY FIT WITHIN BUFFER?
 00000409'EF 02 88 027E 776 BGEQU 20\$: IF GEQU YES
 0000040B'EF 01 90 0285 777 BISB #ERLSM_TIMER,ERL\$GB_BUFFLAG : SET TIMER ACTIVE
 04 A0 08 A0 D0 028C 778 MOVB #1,ERL\$GB_BUFTIM : FORCE ERROR LOG PROCESS WAKE
 00000408'EF 01 8C 0291 779 15\$: MOVL ERL\$L_END(R0),ERL\$L_NEXT(R0) : INDICATE THAT BUFFER IS FULL
 50 00000408'EF 9A 0298 780 XORB #1,ERL\$GB_BUFIND : SWITCH TO ALTERNATE BUFFER
 50 00000400'EF40 D0 029F 781 MOVZBL ERL\$GB_BUFIND,R0 : GET NEW BUFFER INDICATOR
 0B 03 A0 00 E0 02A7 782 MOVL ERL\$AL_BUFADDR[R0],R0 : GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
 52 04 A0 51 C1 02AC 783 BBS #ERLSV_LOCK,ERLSB_FLAGS(R0) : 17\$: IF SET, BUFFER BEING COPIED
 52 08 A0 D1 02B1 784 ADDL3 R1,ERL\$L_NEXT(R0),R2 : CALCULATE ADDRESS OF NEXT AVAILABLE SPACE
 B7 1E 02B5 785 CMPL ERL\$L_END(R0),R2 : ENTRY FIT WITHIN BUFFER?
 04 A0 08 A0 D0 02B7 786 BGEQU 10\$: IF GEQU YES
 50 D4 02BC 787 17\$: MOVL ERL\$L_END(R0),ERL\$L_NEXT(R0) : INDICATE THAT BUFFER IS FULL
 27 11 02BE 788 CLRL R0 : INDICATE ALLOCATION FAILURE
 52 04 C0 02C0 790 20\$: ADDL #EMBSK_LENGTH,R2 : Point past the message header
 62 3E DB 02C3 791 MFPR #PRS_SID,EMBSL HD_SID(R2) : Set system ID into message
 FC A2 51 3C 02C6 792 MOVZWL R1,EMBSW_SIZE(R2) : Set size in message header
 FE A2 02 A0 90 02CA 793 MOVB ERL\$B_BUFIND(R0),EMBSB_BUFIND(R2) : SET RESPECTIVE BUFFER INDICATOR
 60 96 02CF 794 INCB ERL\$B_BUSY(R0) : INCREMENT MESSAGE BUSY COUNT
 51 00000410'EF D0 02D1 795 MOVL ERL\$GC_SEQUENCE,R1 : GET CURRENT ERROR SEQUENCE NUMBER

06 A2 00000000'EF	7D 02D8 796	MOVQ	EXESGQ SYSTIME,EMBSQ,DV,TIME(R2) : INSERT CURRENT TIME
0E A2 51	80 02E0 797	MOVW	R1,EMBSW_DV_ERRSEQ(R2) : INSERT ERROR SEQUENCE NUMBER
50 01	00 02E4 798	MOVL	#1,RO : SET SUCCESS INDICATOR
00000410'EF	D6 02E7 799 30\$:	INCL	ERL\$GL_SEQUENCE : INCREMENT UNIVERSAL ERROR SEQUENCE NUMBER
	02ED 800	ENBINT	
	05 02F0 801	RSB	: ENABLE INTERRUPTS

02F1 803 .SBTTL GET FULL DEVICE NAME
 02F1 804 ;+
 02F1 805 ; ERL\$GETFULLNAME - GET FULL DEVICE NAME
 02F1 806 ;
 02F1 807 ; THIS ROUTINE IS CALLED TO COPY THE FULL DEVICE NAME (NODE NAME + DEVICE NAME)
 02F1 808 ; TO THE ERROR LOG BUFFER.
 02F1 809 ;
 02F1 810 ;
 02F1 811 ;
 02F1 812 ;
 02F1 813 ; R3 = address of DDB
 02F1 814 ; R2 = address of error log buffer
 02F1 815 ;
 02F1 816 ;
 02F1 817 ;
 02F1 818 ; If a node name exist in the system block, it is copied with the
 02F1 819 ; device name to the error log buffer.
 02F1 820 ;
 02F1 821 ;
 02F1 822 ;
 02F1 823 ERL\$GETFULLNAME::
 51 14 A3 9E 02F1 824 MOVAB DDB\$T_NAME(R3),R1 ; Get address of device name.
 7E 81 9A 02F5 825 MOVZBL (R1)+,-(SP) ; Save the string length
 53 34 A3 D0 02F8 826 MOVL DDB\$L_SB(R3),R3 ; Get address of system block
 1A 13 02FC 827 BEQL 208 ; If EQL, go to move device name
 53 44 A3 9E 02FE 828 MOVAB SB\$T_NODENAME(R3),R3 ; Get address of nodename
 50 83 9A 0302 829 MOVZBL (R3)+,R0 ; Get nodename length
 11 13 0305 830 BEQL 208 ; If eql, go move device name
 62 6E 50 81 0307 831 ADDB3 R0,(SP),(R2) ; Nodename length + device name
 82 96 030B 832 INCB (R2)+ ; Total string len. + 1 for '\$'
 d2 83 90 030D 833 10\$: MOV8 (R3)+,(R2)+ ; Copy nodename
 FA 50 F5 0310 834 SOBGTR R0,10\$;
 82 24 90 0313 835 MOVB #^A/\$/, (R2)+ ; Insert the '\$'
 03 11 0316 836 BRB 30\$; Go move device name
 82 6E 90 0318 837 20\$: MOVB (SP),(R2)+ ; Move dev. name len. to buffer
 50 8E D0 031B 838 30\$: MOVL (SP)+,R0 ; Get dev. name length
 82 81 90 031E 839 40\$: MOVB (R1)+,(R2)+ ; Move device name into buffer
 FA 50 F5 0321 840 SOBGTR R0,40\$;
 05 0324 841 RSB ; Return to caller
 0325 842

0325 844 .SBTTL RELEASE ERROR MESSAGE BUFFER
 0325 845 + ERL\$RELEASEMB - RELEASE ERROR MESSAGE BUFFER
 0325 846 THIS ROUTINE IS CALLED TO RELEASE AN ERROR MESSAGE BUFFER FOR PROCESSING
 0325 847 BY THE ERROR LOG PROCESS.
 0325 848
 0325 849
 0325 850
 0325 851
 0325 852
 0325 853
 0325 854
 0325 855 INPUTS:
 0325 856 R2 = ADDRESS OF ERROR MESSAGE BUFFER.
 0325 857
 0325 858
 0325 859
 0325 860
 0325 861
 0325 862
 0325 863
 0325 864
 0325 865 OUTPUTS:
 0325 866 THE COMPLETED ERROR MESSAGE COUNT IS INCREMENTED IN THE RESPECTIVE
 0325 867 ALLOCATION BUFFER HEADER, THE MESSAGE IS SET VALID, AND THE BUSY
 0325 868 MESSAGE COUNT IS DECREMENTED IN THE RESPECTIVE ALLOCATION BUFFER
 0325 869 HEADER.
 0325 870 R3 IS PRESERVED ACROSS CALL.

50 00000400'EF40	50 FF A2 96	0325 866 INCB EMBSB_VALID(R2) ;RELEASE ERROR MESSAGE BUFFER
60 00FF 8F 01	60 00FF 8F 01	0325 867 MOVZBL EMBSB_BUFIND(R2) R0 ;SET MESSAGE BUFFER VALID
00 00000409'EF 01	00 00000409'EF 01	0325 868 MOVL ERL\$AC_BUFADDR[R0] R0 ;GET BUFFER INDICATOR OF ALLOCATION BUFFER
01 A0 0A 07	01 A0 0A 07	0325 869 ADAWI #^XFF, ERL\$B_BUSY(R0) ;GET ADDRESS OF ALLOCATION BUFFER DESCRIPTOR
00000408'EF 01	00000408'EF 01	0325 870 BBCS #ERL\$V_TIMER, ERL\$GB_BUFFLAG, 10\$;ADJUST BUSY AND COMPLETED MESSAGE COUNT
		0325 871 CMPB #MAXMSG, ERL\$B_MSGCNT(R0) ;IF CLR, NO TIMER RUNNING
		0325 872 BGTRU 10\$;MAXIMUM NUMBER OF MESSAGES EXCEEDED?
		0325 873 MOVB #1, ERL\$GB_BUFTIM ;IF GTRU NO
		0325 874 RSB ;FORCE ERROR LOG PROCESS WAKE

10\$: ;

034F 876 .SBTTL WAKE ERROR LOG FORMAT PROCESS
034F 877 .+
034F 878 ERL\$WAKE - WAKE ERROR LOG FORMAT PROCESS
034F 879 .
034F 880 THIS ROUTINE IS CALLED ONCE A SECOND WHEN THE ERROR BUFFER TIMER IS ACTIVE.
034F 881 .
034F 882 INPUTS:
034F 883 .
034F 884 NONE.
034F 885 .
034F 886 OUTPUTS:
034F 887 .
034F 888 THE ERROR BUFFER TIMER IS DECREMENTED AND IF THE RESULT IS ZERO THE
034F 889 ERROR LOG FORMAT PROCESS IS AWAKENED.
034F 890 .-
034F 891 .
034F 892 ERL\$WAKE::: WAKE ERROR LOG FORMAT PROCESS
0000040B'EF 97 034F 893 DECB :DECREMENT TIMER
18 12 0355 894 BNEQ 10\$:
00000409'EF 02 8A 0357 895 BICB #ERLSM_TIMER,ERL\$GB_BUFTIM :CLEAR TIMER ACTIVE FLAG
00000408'EF 1E 90 035E 896 MOVB #MAXTIM,ERL\$GB_BUFTIM :RESET TIMER VALUE
51 0000040C'EF, FC91. D0 0365 897 MOVL ERL\$GL_ERLPID,R1 :GET ERROR LOG PROCESS ID
05 30 036C 898 BSBW SCH\$WARE :WAKE ERROR LOG PROCESS
0370 900 10\$: RSB :
0370 901 .END

ADP_HANDLER
 ADP_UNEXP
 BUFT
 BUF2
 BUGS_UNXINTEXC
 CDDBSB_SYSTEMID
 CDDBSQ_CNTRLID
 CDRPSL_BCNT
 CDRPSL_MEDIA
 CDRPSL_PID
 CDRPSW_BOFF
 CDRPSW_FUNC
 CPU_UNEXP
 DDBSL_SB
 DDBST_NAME
 DDTSL_REGDUMP
 DDTSW_ERRORBUF
 DEV\$V_ELG
 EMBSB_BUFIND
 EMBSB_DV_CLASS
 EMBSB_DV_SLAVE
 EMBSB_DV_TYPE
 EMBSB_LM_CLASS
 EMBSB_LM_TYPE
 EMBSB_SP_CLASS
 EMBSB_SP_TYPE
 EMBSB_VAID
 EMBSC_CS
 EMBSC_DA
 EMBSC_DE
 EMBSC_DT
 EMBSC_LM
 EMBSC_LOGMSCP
 EMBSC_SP
 EMBSC_SU_LENGTH
 EMBSC_UI
 EMBSC_UI_LENGTH
 EMBSC_WS
 EMBSK_HD_LENGTH
 EMBSK_LENGTH
 EMBSK_LM_LENGTH
 EMBSK_SP_LENGTH
 EMBSL_DV_CHAR
 EMBSL_DV_MEDIA
 EMBSL_DV_OPCNT
 EMBSL_DV_OWNUIC
 EMBSL_DV_REGS
 EMBSL_DV_RQPID
 EMBSL_HD_SID
 EMBSL_SP_BCN
 EMBSL_SP_CHAR
 EMBSL_SP_CMDREF
 EMBSL_SP_MEDIA
 EMBSL_SP_OPCNT
 EMBSL_SP_OWNUIC
 EMBSL_SP_RQPID
 EMBSL_SU_DAYT

0000003E	R	03	EMBSL_UI_CSR	= 00000014
00000000	RR	03	EMBSL_UI_TR	= 00000010
00000000	RR	02	EMBSQ_DV_IOSB	= 00000012
00000200	R	02	EMBSQ_DV_TIME	= 00000006
*****	X	03	EMBSQ_SP_IOSB	= 00000020
= 0000000C			EMBSET_DV_NAME	= 0000003E
= 00000020			EMBSET_LM_DEVNAM	= 00000014
= FFFFFFD2			EMBSET_SP_DEVNAM	= 00000040
= FFFFFFD8			EMBSW_DV_BCNT	= 00000024
= FFFFFFAC			EMBSW_DV_BOFF	= 00000022
= FFFFFFD0			EMBSW_DV_ENTRY	= 00000004
= FFFFFFC0			EMBSW_DV_ERRCNT	= 0000002C
00000088	R	03	EMBSW_DV_ERRSEQ	= 0000000E
= 00000034			EMBSW_DV_FUNC	= 0000003C
= 00000014			EMBSW_DV_STS	= 0000001A
= 00000010			EMBSW_DV_UNIT	= 0000002A
= 00000016			EMBSW_HD_ENTRY	= 00000004
= 00000016			EMBSW_LM_ENTRY	= 00000004
= FFFFFFFE			EMBSW_LM_MSGTYP	= 00000024
= 0000001C			EMBSW_LM_UNIT	= 00000012
= 0000003A			EMBSW_SIZE	= FFFFFFFC
= 0000001D			EMBSW_SP_BOFF	= 00000012
= 00000010			EMBSW_SP_ENTRY	= 00000004
= 00000011			EMBSW_SP_ERRCNT	= 00000030
= 00000010			EMBSW_SP_FUNC	= 00000028
= 00000011			EMBSW_SP_JCBSTS	= 00000032
= FFFFFFFF			EMBSW_SP_UNIT	= 0000002A
= 00000020			EMBSW_SU_ENTRY	= 00000004
= 00000062			EMBSW_UI_ENTRY	= 00000004
= 00000001			ERL\$ALLOCMB	00000251 RG 04
= 00000060			ERL\$AL_BUFADDR	00000400 RG 02
= 00000064			ERL\$B_BUFIND	= 00000002
= 00000065			ERL\$B_BUSY	= 00000000
= 00000063			ERL\$B_FLAGS	= 00000003
= 00000014			ERL\$B_MSGCNT	= 00000001
= 00000061			ERL\$COLDSTART	0000022F RG 04
= 00000018			ERL\$C_LENGTH	= 0000000C RG 04
= 00000024			ERL\$DEVICEATTN	00000096 RG 04
= 00000010			ERL\$DEVICERR	00000000 RG 04
= 00000004			ERL\$DEVICTMO	00000004 RG 04
= 00000026			ERL\$GB_BUFFLAG	00000409 RG 02
= 00000050			ERL\$GB_BUFIND	00000408 RG 02
= 00000036			ERL\$GB_BUFPTR	0000040A RG 02
= 00000026			ERL\$GB_BUFTIM	0000040B RG 02
= 0000002E			ERL\$GETFULLNAME	000002F1 RG 04
= 00000032			ERL\$GL_ERLPID	0000040C RG 02
= 0000004E			ERL\$GL_SEQUENCE	00000410 RG 02
= 0000001E			ERL\$LOGMESSAGE	00000189 RG 04
= 00000000			ERL\$LOGSTATUS	0000010E RG 04
= 00000014			ERL\$LOG_DMSCP	000001F3 RG 04
= 00000038			ERL\$LOG_TMSCP	000001E8 RG 04
= 0000003C			ERL\$L_END	= 00000008
= 00000018			ERL\$L_NEXT	= 00000004
= 0000002C			ERL\$M_TIMER	= 00000002
= 00000034			ERL\$RELEASEMB	00000325 RG 04
= 0000001C			ERL\$UNEXP	00000088 RG 03
= 00000010			ERL\$VECO	00000088 RG 03

ERL\$VEC100
ERL\$VEC104
ERL\$VEC108
ERL\$VEC112
ERL\$VEC116
ERL\$VEC12
ERL\$VEC120
ERL\$VEC124
ERL\$VEC128
ERL\$VEC132
ERL\$VEC136
ERL\$VEC140
ERL\$VEC144
ERL\$VEC148
ERL\$VEC152
ERL\$VEC156
ERL\$VEC16
ERL\$VEC160
ERL\$VEC164
ERL\$VEC168
ERL\$VEC172
ERL\$VEC176
ERL\$VEC180
ERL\$VEC184
ERL\$VEC188
ERL\$VEC192
ERL\$VEC196
ERL\$VEC20
ERL\$VEC200
ERL\$VEC204
ERL\$VEC208
ERL\$VEC212
ERL\$VEC216
ERL\$VEC220
ERL\$VEC224
ERL\$VEC228
ERL\$VEC232
ERL\$VEC236
ERL\$VEC24
ERL\$VEC240
ERL\$VEC244
ERL\$VEC248
ERL\$VEC252
ERL\$VEC256
ERL\$VEC260
ERL\$VEC264
ERL\$VEC268
ERL\$VEC272
ERL\$VEC276
ERL\$VEC28
ERL\$VEC280
ERL\$VEC284
ERL\$VEC288
ERL\$VEC292
ERL\$VEC296
ERL\$VEC300
ERL\$VEC304

000000B8 RG 03	ERL\$VEC308	00000034 RG 03
000000B8 RG 03	ERL\$VEC312	00000038 RG 03
000000B8 RG 03	ERL\$VEC316	0000003C RG 03
000000B8 RG 03	ERL\$VEC32	000000B8 RG 03
000000B8 RG 03	ERL\$VEC320	00000000 RG 03
000000B8 RG 03	ERL\$VEC324	00000004 RG 03
000000B8 RG 03	ERL\$VEC328	00000008 RG 03
000000B8 RG 03	ERL\$VEC332	0000000C RG 03
000000B8 RG 03	ERL\$VEC336	00000010 RG 03
000000B8 RG 03	ERL\$VEC340	00000014 RG 03
000000B8 RG 03	ERL\$VEC344	00000018 RG 03
000000B8 RG 03	ERL\$VEC348	0000001C RG 03
000000B8 RG 03	ERL\$VEC352	00000020 RG 03
000000B8 RG 03	ERL\$VEC356	00000024 RG 03
000000B8 RG 03	ERL\$VEC36	000000B8 RG 03
000000B8 RG 03	ERL\$VEC360	00000028 RG 03
000000B8 RG 03	ERL\$VEC364	0000002C RG 03
000000B8 RG 03	ERL\$VEC368	00000030 RG 03
000000B8 RG 03	ERL\$VEC372	00000034 RG 03
000000B8 RG 03	ERL\$VEC376	00000038 RG 03
000000B8 RG 03	ERL\$VEC380	0000003C RG 03
000000B8 RG 03	ERL\$VEC384	00000000 RG 03
000000B8 RG 03	ERL\$VEC388	00000004 RG 03
000000B8 RG 03	ERL\$VEC392	00000008 RG 03
000000B8 RG 03	ERL\$VEC396	0000000C RG 03
000000B8 RG 03	ERL\$VEC4	000000B8 RG 03
000000B8 RG 03	ERL\$VEC40	000000B8 RG 03
000000B8 RG 03	ERL\$VEC400	00000010 RG 03
000000B8 RG 03	ERL\$VEC404	00000014 RG 03
000000B8 RG 03	ERL\$VEC408	00000018 RG 03
000000B8 RG 03	ERL\$VEC412	0000001C RG 03
000000B8 RG 03	ERL\$VEC416	00000020 RG 03
000000B8 RG 03	ERL\$VEC420	00000024 RG 03
000000B8 RG 03	ERL\$VEC424	00000028 RG 03
000000B8 RG 03	ERL\$VEC428	0000002C RG 03
000000B8 RG 03	ERL\$VEC432	00000030 RG 03
000000B8 RG 03	ERL\$VEC436	00000034 RG 03
000000B8 RG 03	ERL\$VEC44	000000B8 RG 03
000000B8 RG 03	ERL\$VEC440	00000038 RG 03
000000B8 RG 03	ERL\$VEC444	0000003C RG 03
000000B8 RG 03	ERL\$VEC448	00000000 RG 03
000000B8 RG 03	ERL\$VEC452	00000004 RG 03
000000B8 RG 03	ERL\$VEC456	00000008 RG 03
00000000 RG 03	ERL\$VEC460	0000000C RG 03
00000004 RG 03	ERL\$VEC464	00000010 RG 03
00000008 RG 03	ERL\$VEC468	00000014 RG 03
0000000C RG 03	ERL\$VEC472	00000018 RG 03
00000010 RG 03	ERL\$VEC476	0000001C RG 03
00000014 RG 03	ERL\$VEC48	000000B8 RG 03
000000B8 RG 03	ERL\$VEC480	00000020 RG 03
00000018 RG 03	ERL\$VEC484	00000024 RG 03
0000001C RG 03	ERL\$VEC488	00000028 RG 03
00000020 RG 03	ERL\$VEC492	0000002C RG 03
00000024 RG 03	ERL\$VEC496	00000030 RG 03
00000028 RG 03	ERL\$VEC500	00000034 RG 03
0000002C RG 03	ERL\$VEC504	00000038 RG 03
00000030 RG 03	ERL\$VEC508	0000003C RG 03

ERL\$VEC52	000000B8	RG
ERL\$VEC56	000000B8	RG
ERL\$VEC60	000000B8	RG
ERL\$VEC64	000000B8	RG
ERL\$VEC68	000000B8	RG
ERL\$VEC72	000000B8	RG
ERL\$VEC76	000000B8	RG
ERL\$VEC8	000000B8	RG
ERL\$VEC80	000000B8	RG
ERL\$VEC84	000000B8	RG
ERL\$VEC88	000000B8	RG
ERL\$VEC92	000000B8	RG
ERL\$VEC96	000000B8	RG
ERL\$VEC_RETURN	000000C0	RG
ERL\$V_LOCK	= 00000000	
ERL\$V_TIMER	= 00000001	
ERL\$WAKE	= 000034F	RG
ERL\$WARMSTART	= 0000234	RG
EXESGL_CONFREGL	*****	X
EXESGQ_SYSTIME	*****	X
EXESMC4K_PRTCT	*****	X
EXESREAD_TODR	*****	X
IOSGL_SCB_INTO	*****	X
IOSV_INHERLOG	= 0000000B	
IRPSL_PID	= 0000000C	
IRPSW_BOFF	= 00000030	
IRPSW_FUNC	= 00000020	
MAXMSG	= 0000000A	
MAXTIM	= 0000001E	
MCHKSM_LOG	= 00000001	
MCHKSM_NEXM	= 00000004	
MMGSL_SBICONF	*****	X
MSCPSL_CMD_REF	= 00000000	
NDTS_DR32	= 00000030	
NEXUS	= 00000010	
ORBSL_OWNER	= 00000000	
PRS_IPL	= 00000012	
PRS_SID	= 0000003E	
SBST_NODENAME	= 00000044	
SCHSHAKE	*****	X
UCBSB_DEVCLASS	= 00000040	
UCBSB_DEVTYPE	= 00000041	
UCBSB_SLAVE	= 00000090	
UCBSL_DDB	= 00000028	
UCBSL_DDT	= 00000088	
UCBSL_DEVCHAR	= 00000038	
UCBSL_EMB	= 00000094	
UCBSL_IRP	= 00000058	
UCBSL_MEDIA	= 000000BC	
UCBSL_OPCNT	= 00000070	
UCBSL_ORB	= 0000001C	
UCBSM_ERLOGIP	= 00000004	
UCBSV_ERLOGIP	= 00000002	
UCBSW_ERRCNT	= 00000082	
UCBSW_FUNC	= 0000009A	
UCBSW_STS	= 00000064	
UCBSW_UNIT	= 00000054	

```
+-----+
! Psect synopsis !
+-----+
```

PSECT name

	Allocation	PSECT No.	Attributes																	
: ABS .	00000000	(0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE						
\$ABSS	00000000	(0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE						
SS\$260	00000414	(1044.)	02 (2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	QUAD						
SAEXENONPAGED	000000C7	(199.)	03 (3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	LONG						
WIONONPAGED	00000370	(880.)	04 (4.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE						

```
+-----+
! Performance indicators !
+-----+
```

Phase

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.07	00:00:01.74
Command processing	108	00:00:00.49	00:00:04.75
Pass 1	549	00:00:23.65	00:01:09.35
Symbol table sort	0	00:00:03.40	00:00:11.63
Pass 2	174	00:00:04.51	00:00:14.71
Symbol table output	34	00:00:00.28	00:00:01.92
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	905	00:00:32.44	00:01:44.15

The working set limit was 1950 pages.

130939 bytes (256 pages) of virtual memory were used to buffer the intermediate code.

There were 120 pages of symbol table space allocated to hold 2271 non-local and 34 local symbols.

901 source lines were read in Pass 1, producing 25 object records in Pass 2.

41 pages of virtual memory were used to define 40 macros.

```
+-----+
! Macro library statistics !
+-----+
```

Macro library name

Macro library name	Macros defined
\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	28
\$255\$DUA28:[SYSLIB]STARLET.MLB;2	7
TOTALS (all libraries)	35

2304 GETS were required to define 35 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LI\$:\$ERRORLOG/OBJ=OBJ\$:\$ERRORLOG MSRC\$:\$ERRORLOG/UPDATE=(ENH\$:\$ERRORLOG)+EXECMLS/LIB

0374 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DISMOUNT
LIS

DEBUGDATA
LIS

ERRORLOG
LIS

DEVICECDAT
LIS

EXCEPTION
LIS

EXCEPMSG
LIS

FILEREAD
LIS

EXSUBROUT
LIS

FILERWIO
LIS